Energy, Mines and Resources Canada
Énergie, Mines et Ressources Canada

CANMET

Canada Centre for Mineral and Energy Technology
Centre canadien de la technologie des minéraux et de l'énergie

IMPRESSLIB
Fortran Impress Interface Library

N. Toews and A.S. Wong
Canadian Mine Technology Laboratory

MARCH 1987

This document is an unedited
author's report prepared for the
use of the Canada Centre
for Mineral and Energy Technology
(CANMET)

Ce document est un rapport
préparé pour les besoins du Centre
canadien de la technologie des minéraux et de
l'énergie (CANMET)

22 pp

MICROMEDIA

MINING RESEARCH LABORATORIES
DIVISIONAL REPORT MRL 87-25 (TR)

IMPRESSLIB

Fortran Impress Interface Library

by

N. Toews* and A. S. Wong**

ABSTRACT

This report describes a library of fortran callable routines. This subroutine library makes it possible for users to control the Imagen laser printer directly. Both textual and graphic output can be produced.

Keywords — Program, Fortran, Laser, Graphics, Interface Library.

---

* Research Scientist, Mining Research Laboratories, CANMET, Energy, Mines and Resources, Ottawa, Canada.

** Physical Scientist, Mining Research Laboratories, CANMET, Energy, Mines and Resources, Ottawa, Canada.

IMPRESSLIB

Bibliothèque d'interface Impress FORTRAN

par

N. Toews* et A. S. Wong**

RÉSUMÉ

Le présent rapport décrit une bibliothèque de programmes d'appel FORTRAN. Grâce à cette bibliothèque de sous-programmes, les utilisateurs peuvent commander directement l'imprimante à laser Imagen. Le texte et les graphiques peuvent être produits.

MOTS-CLÉS : Programme, FORTRAN, laser, graphiques, bibliothèque d'interface.

---

* Chercheur scientifique, Laboratoires de recherche minière, CANMET, Énergie, Mines et Ressources Canada, Ottawa, Canada.

** Chercheur en sciences physiques, Laboratoires de recherche minière, CANMET, Énergie, Mines et Ressources Canada, Ottawa, Canada.

CONTENTS

	Page No.
ABSTRACT	i
RÉSUMÉ	ii
INTRODUCTION	1
USING THE INTERFACE LIBRARY	1
FORTRAN IMPRESS INTERFACE ROUTINES	2
CONTROL ROUTINES	2
IMPOPN	2
IMPDMP	2
ENDPAGE	2
PHYSICAL AND LOGICAL AXES	2
SET_HV_SYSTEM(ORIGIN,AXES,ORIENTATION)	2
POSITIONING	3
SET_ABS_H(NEW_H) and SET_ABS_V(NEW_V)	3
SET_REL_H(DELTA_H) and SET_REL_V(DELTA_V)	3
GRAPHICS ROUTINES	3
LOAD_TEXTURE(FAMILY,MEMBER,TEXTURE_NO)	3
SET_TEXTURE(FAMILY,MEMBER)	5
CREATE_PATH(COUNT,H,V)	5
SET_PUM(MODE)	5
DRAW_PATH(OPER)	5
SET_PEN(DIAMETER)	8
FILL_PATH(OPER)	8
TEXT OUTPUT ROUTINES	10
SET_ADV_DIRS(MAIN,SECONDARY)	10
LOAD_FONT(FAMILY,FONT)	10
SET_FAMILY(FAMILY)	12
SET_SP(SPACE_WIDTH)	12
SET_BOL(LINE_BEGIN)	12
SET_IL(INTER_LINE)	12
MMOVE(DELTA_M)	12
SMOVE(DELTA_S)	12
LINE_TEXT(STRING)	13
CHARS_TEXT(STRING)	13
EXAMPLES	13

CONTENTS
(Continued)

Page No.

FIGURES

Figure 1. Logical Coordinate Systems 4
Figure 2. Predefined Textures 6
Figure 3. Pie Chart with Textures 7
Figure 4. Pen Diameters 8
Figure 5. Clarification of "OPER" 9
Figure 6. Laser Printer Resident Fonts 11
Figure 7. Graphical Fortran Demo Program 14
Figure 8. Output of Graphical Demo Program 15
Figure 9. Text Fortran Demo Program 16
Figure 10. Output of Text Demo Program 17

TABLES

Table 1. Space Width for Resident Fonts 11

INTRODUCTION

MRL has an Imagen 8/300 laser printer. The language understood by this printer is called "IMPRESS". To fully access all the capabilities of the laser printer, both graphics and text, it is necessary to use "IMPRESS". "IMPRESS" is a machine language with no high level language interface available. This report describes a fortran interface library, written at MRL, that accesses almost all of IMPRESS's capabilities.

TEX is a text formatting language extensively used at MRL. TEX permits the direct insertion of IMPRESS files. This allows graphics to be merged with text, in reports and memos.

The fortran library, "IMPRESSLIB", described in this report has been in use at MRL for 2 years. "TEKIMP", the program that converts tektronix graphical output to IMPRESS, was written using this library. The maple leaf, on the report cover, was created using "IMPRESSLIB".

IMPRESS has no predefined textures or patterns to use in area fill operations. A number of textures have been developed at MRL. These are included in the fortran library and are described in this report.

USING THE FORTRAN IMPRESS INTERFACE LIBRARY

The user creates a fortran program (e.g. myprog.for), which calls various fortran/-impress routines. This is then compiled:

```
$for myprog
```

This is then linked:

```
$link myprog,impresslib/lib
```

The "impress" file produced when myprog.exe is run is stored on a file with logical name "impout". The user can assign any filename to "impout":

```
$define impout myprog.imp
$run myprog
```

The user then submits myprog.imp to "imprint". "Imprint" queues the file for printing on the laser printer.

```
$imp/noreverse myprog.imp
```

Note that imprint realizes that the file is an impress file from the file extension ".imp". If "impout" had been assigned to myprog.dat, for instance, then it would be necessary to submit myprog.dat to imprint as follows:

```
$imp/impress/noreverse myprog.dat
```

FORTRAN IMPRESS INTERFACE ROUTINES

CONTROL ROUTINES

IMPOPN

The first impress routine called must be **IMPOPN**. This opens the impress output file with logical name "impout".

IMPDMP

IMPDMP is the last impress routine called. It empties the internal buffer by writing it to file "impout".

ENDPAGE

ENDPAGE tells the laser printer that the current page is complete. A new page will be started if there is additional output.

PHYSICAL AND LOGICAL AXES

The physical coordinate system is called the (x,y) system, and is at the upper left hand corner of the page (longside of page vertical). The x-axis is to the right and the y-axis is down. This is shown in figure 1.

The logical coordinate axes, called the (h,v) system, is the system within which the user defines coordinates. The user can define this system anywhere on the printer page. The default logical (h,v) system coincides with the physical (x,y) system.

The laser coordinates of a point on the printer page, are the number of laser dots measured from the origin in the current (h,v) system. The number of dots per inch in the physical x-direction is approximately 286 and the number of dots per inch in the physical y-direction is approximately 303.

SET_HV_SYSTEM

The (h,v) system used, can be redefined by a call to "SET_HV_SYSTEM".

SET_HV_SYSTEM(ORIGIN,AXES,ORIENTATION)

where

ORIGIN — an integer with possible values 0 to 3
=0 or 1, no change.
=2, set origin to "top left corner".
=3, set origin to current location.

AXES — an integer with possible values 0 to 3.
=0, no change.
=1, invert the relationship of h and v axes.
=2, h to v is 90 degrees clockwise.
=3, h to v is 90 degrees counterclockwise.

ORIENTATION — an integer with possible values 0 to 7. It defines a new h-axis.
=0, 0 degrees from current h-axis (no change).
=1, 90 degrees from current h-axis.
=2, 180 degrees from current h-axis.
=3, 270 degrees from current h-axis.
=4, 0 degrees from physical x-axis.
=5, 90 degrees from physical x-axis.
=6, 180 degrees from physical x-axis.
=7, 270 degrees from physical x-axis.

Figure 1 shows some possible (h,v) logical systems.

POSITIONING

SET_ABS_H(NEW_H) and **SET_ABS_V(NEW_V)**

Move to the point (**NEW_H**,**NEW_V**) in the current logical coordinate system.

SET_REL_H(DELTA_H) and **SET_REL_V(DELTA_V)**

Move from current point a relative distance (**DELTA_H**,**DELTA_V**).

GRAPHICS ROUTINES

The user specifies a polygonal line using "**CREATE_PATH**". This line can then be drawn, with specified width, using "**DRAW_PATH**". Alternatively, the inside of the region defined can be filled, using "**FILL_PATH**". The line or fill can be black, or can be any of the predefined textures.

LOAD_TEXTURE(FAMILY,MEMBER,TEXTURE_NO)

This causes a texture to be defined and made available to the laser printer.
Arguments

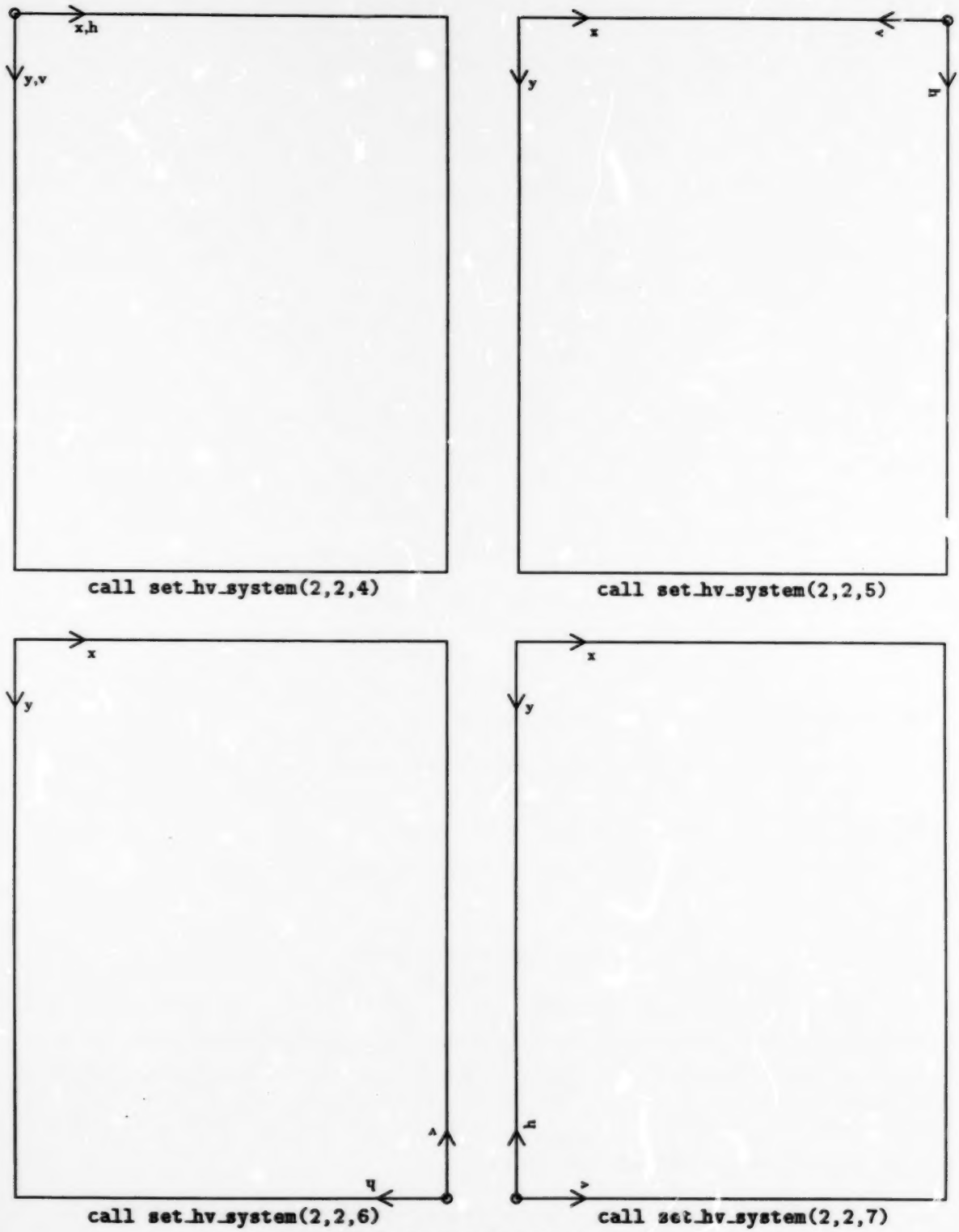

FIGURE 1. Logical Coordinate Systems

FAMILY — is an integer in the range 0 to 95.

MEMBER — is an integer in the range 1 to 127.

TEXTURE_NO — is and integer specifying the texture. Currently the maximum is 15. The various textures available, with associated texture number is shown in figure 2.

Note that a (FAMILY, MEMBER) identifier cannot be redefined once it has been used. There can only be one current texture. This is defined by calling the following routine.

SET_TEXTURE(FAMILY,MEMBER)

This sets the current texture. The (FAMILY,MEMBER) must have been previously defined by a call to "LOAD_TEXTURE".

CREATE_PATH(COUNT,H,V)

This command defines a polygonal path by the coordinates (H(1),V(1)), to (H(COUNT),V(COUNT)). The coordinates are relative to the current logical coordinate system.

arguments

COUNT — an integer specifying the number of points.

H,V — integer arrays defining the point coordinates.

A path can be self intersecting. If a fill operation is to be carried out it is not necessary that the first and last points coincide. Impress will join the first and last points if necessary.

Successive calls to "CREATE_PATH" will replace the old path with the new path, unless the update mode is set to "append". If the update mode is "append", then a new path is created by joining the last point of the old path to the first point of the new path.

SET_PUM(MODE)

This routine sets the current path update mode.

MODE — an integer

=0, no append (old path is discarded). This is the default.

=1, update mode is append.

DRAW_PATH(OPER)

This routine draws a polygonal line, of specified width, on the path defined by "CREATE_PATH".

OPER — an integer

=0, causes a white line to be drawn. Note that this

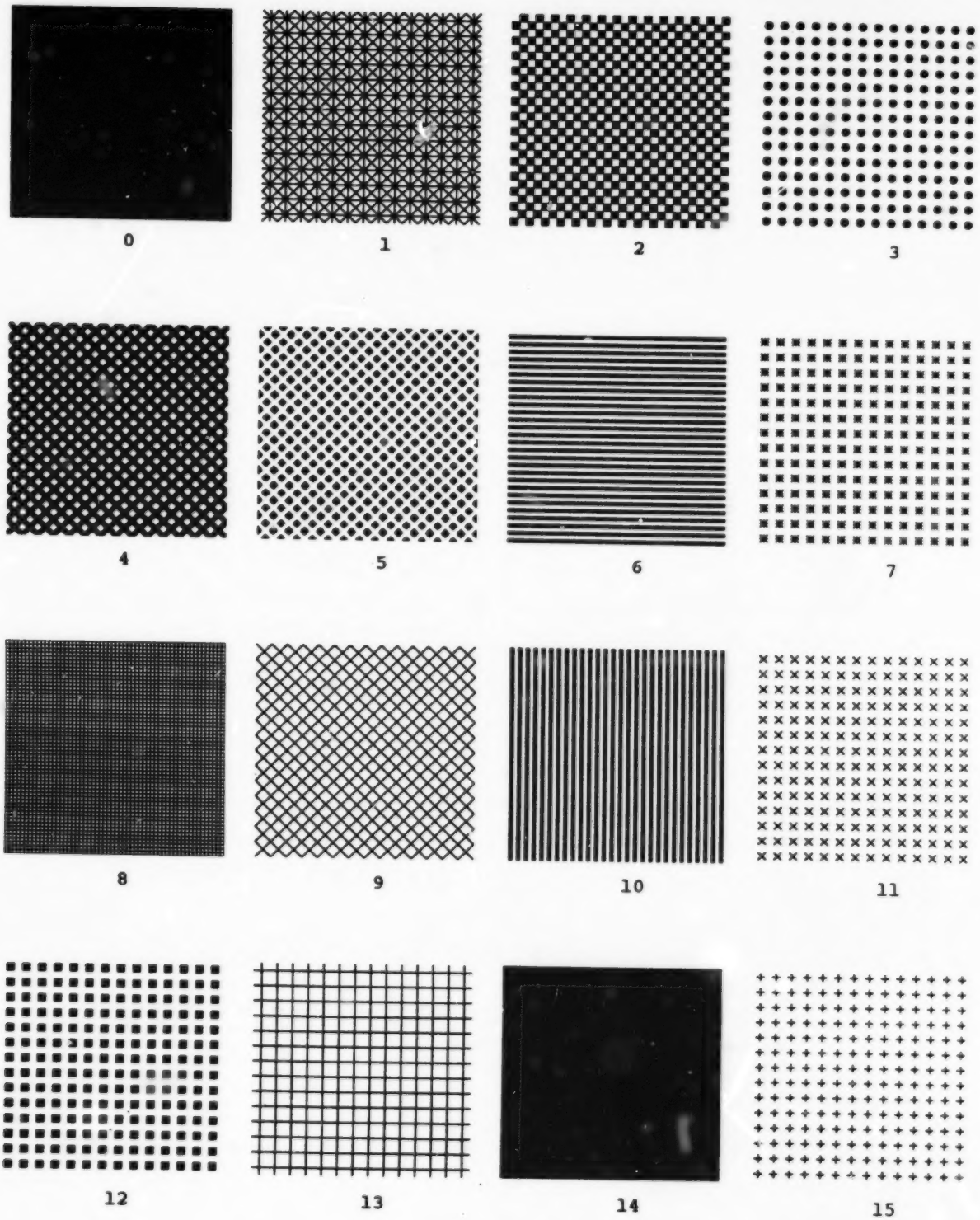

FIGURE 2: Predefined Textures

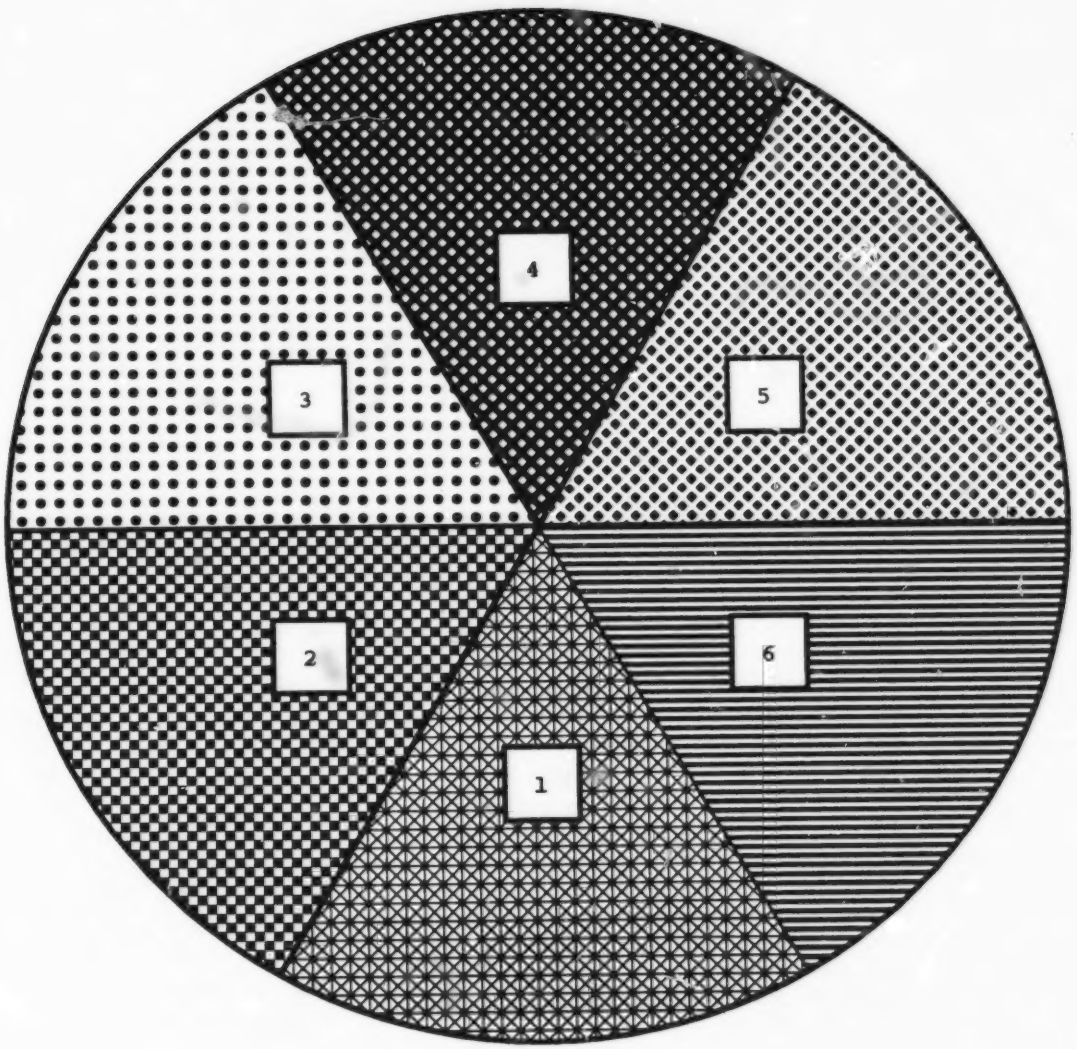

FIGURE 3: Pie Chart with Textures

will not be visible unless previous graphics or text output is being overwritten.

=3, in this case the current texture is used. Existing output is overwritten.

=7, An "or" operation between the current texture and what is already on the page is carried out. i.e. old output is not destroyed.

=15, causes a black line to be drawn.

Line width is defined by a call to "SET_PEN".

SET_PEN(DIAMETER)

DIAMETER — an integer in the range 1 to 20. The default is 1. The width is specified in laser coordinates. See Figure 4.

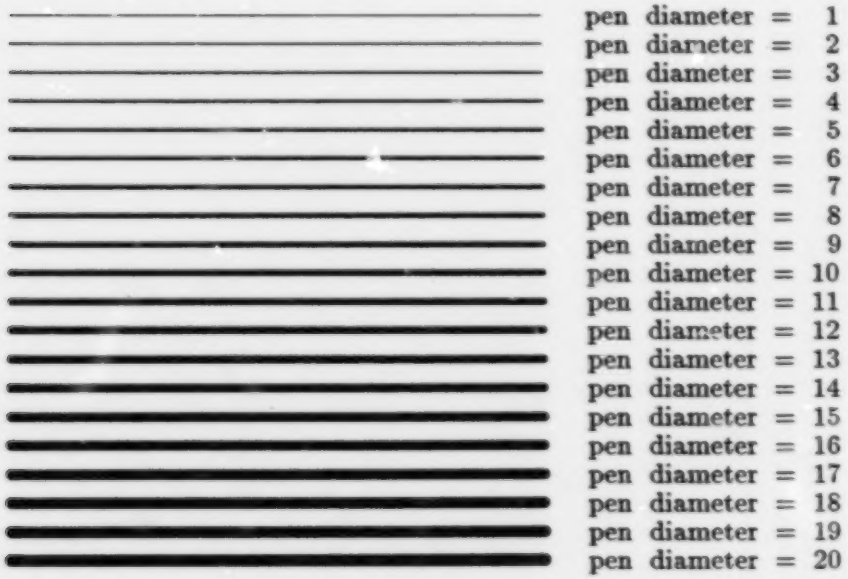
pen diameter = 1
pen diameter = 2
pen diameter = 3
pen diameter = 4
pen diameter = 5
pen diameter = 6
pen diameter = 7
pen diameter = 8
pen diameter = 9
pen diameter = 10
pen diameter = 11
pen diameter = 12
pen diameter = 13
pen diameter = 14
pen diameter = 15
pen diameter = 16
pen diameter = 17
pen diameter = 18
pen diameter = 19
pen diameter = 20

FIGURE 4: Pen Diameters

FILL_PATH(OPER)

The operation fills the polygon defined by the current path. If necessary, the first and last points are joined.

OPER — an integer

=0, causes white fill to be used. Note that this will not be visible unless previous graphics or text output is being overwritten.

=3, in this case the current texture is used as fill. Existing output is overwritten.

=7, An "or" operation between the current texture and what is already on the page is carried out. i.e. old output is not destroyed.

=15, causes black fill to be used.

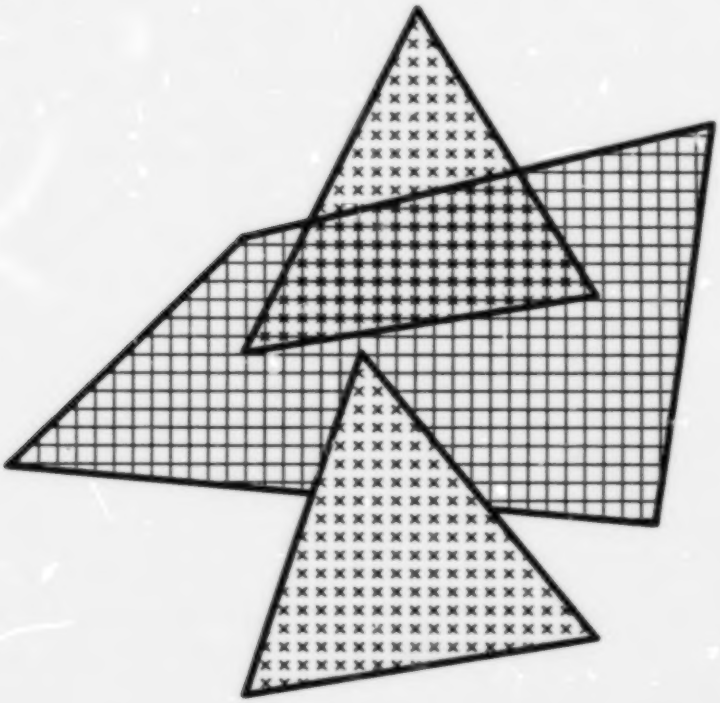

FIGURE 5: Clarification of "OPER"

Figure 5 clarifies the meaning of "OPER". The filled quadrilateral, with black boundary and texture#13, was produced first. Then the two triangles,both with texture#11, were created. The top triangle was created with "OPER=7" and the lower triangle with "OPER=3".

TEXT OUTPUT ROUTINES

Text output is constrained to lines parallel to the page edges. The direction of a line of characters, called the main axis, can only have one of four directions (up, down, left, right). Given the main axis, then the direction in which additional lines are inserted, called the secondary axis, is perpendicular to the main axis. It can be either direction. The default main axis is in the direction of increasing logical "h", and the default secondary axis is in the direction of increasing logical "v". The user can define other (m,s) axes by calling "SET_ADV_DIRS".

```
SET_ADV_DIRS(MAIN,SECONDARY)
```

where

MAIN — an integer with possible values 0,1,2 or 3.
=0, 0 degrees from current h-axis.
=1, 90 degrees from current h-axis
=2, 180 degrees from current h-axis
=3, 270 degrees from current h-axis

SECONDARY — an integer with possible values 0 or 1.
=0, clockwise 90 degrees from the m-axis.
=1, counterclockwise 90 degrees from the m-axis.

All the resident fonts on the laser printer are available for use. See figure 6. It is also possible to download the TEX fonts from the Vax. A font is made ready for use, by calling the routine "LOAD_FONT".

```
LOAD_FONT(FAMILY,FONT)
```

where

FAMILY — an integer from 0 to 95. Note a family,
once defined, cannot be redefined.

FONT — a character string defining the font.
e.g. `CALL LOAD_FONT(1,'cour10')`
loads resident font "cour10".
`CALL LOAD_FONT(2,'cmr10.1200')`
loads TEX font "cmr10" magnified "magstep1".

Before text output can be produced, one of the loaded fonts must be defined as the current font. This is done by a call to routine "SET_FAMILY".

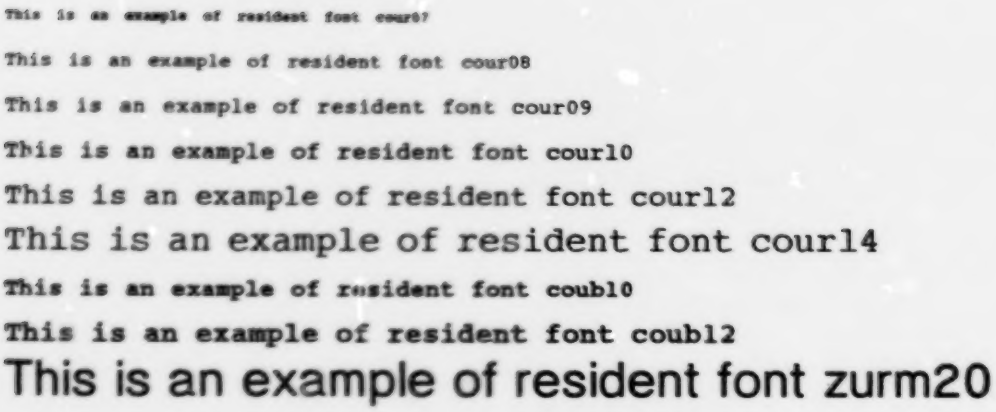
This is an example of resident font cour07
This is an example of resident font cour08
This is an example of resident font cour09
This is an example of resident font courl0
This is an example of resident font courl2
This is an example of resident font courl4
This is an example of resident font coubl0
This is an example of resident font coubl2
This is an example of resident font zurm20

FIGURE 6. Laser Printer Resident Fonts

TABLE 1

Space Width For Resident Fonts

Resident Font	Space
cour07	15
cour08	20
cour09	23
courl0	25
courl2	30
courl4	37
coubl0	25
coubl2	30
zurm20	55

SET_FAMILY(FAMILY)

where

FAMILY — an integer. This must have been previously associated with a font.

There is no space character in the fonts. The user must define the width of the space character by calling "**SET_SP**".

SET_SP(SPACE_WIDTH)

where

SPACE_WIDTH — an integer specifying the space width in laser units. Table 1 shows appropriate values to use with resident fonts.

A space is created by a call to "**SP**", **CALL SP**. This uses the space width previously defined by a call to "**SET_SP**".

The location of the beginning of the line (the left margin) and the interline spacing are set by calls to "**SET_BOL**" and "**SET_IL**" respectively.

SET_BOL(LINE_BEGIN)

where

LINE_BEGIN —an integer defining the beginning of line in laser coordinates from the current logical origin.

SET_IL(INTER_LINE)

where

INTER_LINE — an integer defining the interline spacing in laser coordinates.

A call to "**CRLF**", (**CALL CRLF**), causes the laser to position itself at the beginning of the next line using the values specified by "**LINE_BEGIN**" and "**INTER_LINE**".

Movement in the main and secondary directions can be achieved by calls to "**MMOVE**" and "**SMOVE**" respectively.

MMOVE(DELTA_M)

where

DELTA_M — an integer specifying an incremental move along the main axis, in laser coordinates.

SMOVE(DELTA_S)

where

DELTA_S — an integer specifying an incremental move along the secondary axis, in laser coordinates.

Two routines, "LINE_TEXT" and "CHARS_TEXT", are used to output text data.

LINE_TEXT(STRING)

where

STRING — a string of characters that are to be output at the current location

Blank characters in the string "STRING" cause a call to "SP". At the end of the string, "CRLF" is called automatically.

"CHARS_TEXT" is identical to "LINE_TEXT" except that there is no automatic call to "CRLF", at the end of the string. This allows font changes within a line, etc.

EXAMPLES

Two fortran demo programs with output are shown. Figure 7 shows an example of a fortran program using graphical commands. Figure 8 displays the output from this example. Figure 9 shows and example of a fortran program using text commands. Figure 10 displays the output from this example.

```
      PROGRAM GRAF
C     TO DEMONSTRATE GRAPHICS COMMANDS
      INTEGER H(5),V(5),QH(5),QV(5)
C     DEFINE A QUADRILATERAL
      DATA QH /-200,-100,300,200,-200/
      DATA QV /200,-100,-300,200,200/
      CALL IMPOPN
C     TRANSLATE QUAD AND DRAW WITH SOLID LINE
      DO 10 I=1,5
      H(I)=QH(I)+1200
   10 V(I)=QV(I)+800
      CALL SET_PEN(5)
      CALL CREATE_PATH(5,H,V)
      CALL DRAW_PATH(15)
C     TRANSLATE QUAD AND FILL WITH BLACK
      DO 20 I=1,5
      H(I)=QH(I)+1200
   20 V(I)=QV(I)+1500
      CALL CREATE_PATH(5,H,V)
      CALL FILL_PATH(15)
C     TRANSLATE QUAD AND FILL WITH TEXTURE 2
      DO 30 I=1,5
      H(I)=QH(I)+1200
   30 V(I)=QV(I)+2200
      CALL LOAD_TEXTURE(1,1,2)
      CALL SET_TEXTURE(1,1)
      CALL CREATE_PATH(5,H,V)
      CALL FILL_PATH(3)
C     ADD BLACK BORDER
      CALL SET_PEN(10)
      CALL DRAW_PATH(15)
      CALL ENDPAGE
      CALL IMPDMP
      STOP
      END
```

FIGURE 7: Graphical Fortran Demo Program

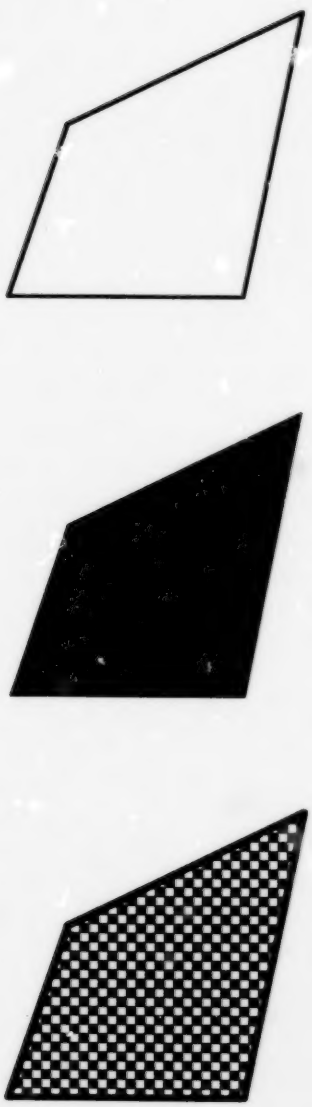

FIGURE 8: Output of Graphical Demo Program

```
      PROGRAM TEXT
C     TO DEMONSTRATE TEXT COMMANDS
      CALL IMPOPN
C     LOAD RESIDENT FONT COUR10
      CALL LOAD_FONT(1,'COUR10')
C     LOAD SOME TEX FONTS
      CALL LOAD_FONT(2,'CMR10.1200')
      CALL LOAD_FONT(3,'CMBX10.1200')
      CALL LOAD_FONT(4,'CMTI10.1200')
      CALL LOAD_FONT(5,'CMINCH.1000')
C     SET LEFT MARGIN AND LINE SPACING
      CALL SET_BOL(300)
      CALL SET_IL(80)
C     SET SPACE WIDTH
      CALL SET_SP(28)
      CALL SET_ABS_H(300)
      CALL SET_ABS_V(300)
C     OUTPUT TEXT
      CALL SET_FAMILY(1)
      CALL LINE_TEXT('Text output using resident cour10 font')
      CALL SET_FAMILY(2)
      CALL LINE_TEXT('Text output using TEX font cmr10.1200')
      CALL CHARS_TEXT('There is a')
      CALL SET_FAMILY(3)
      CALL CHARS_TEXT(' bold')
      CALL SET_FAMILY(2)
      CALL CHARS_TEXT(' and an')
      CALL SET_FAMILY(4)
      CALL CHARS_TEXT(' italic')
      CALL SET_FAMILY(2)
      CALL CHARS_TEXT(' in this sentence')
      CALL SET_ADV_DIRS(1,1)
      CALL SET_ABS_H(900)
      CALL SET_ABS_V(525)
      CALL SET_FAMILY(5)
      CALL LINE_TEXT('GOODBYE')
      CALL ENDPAGE
      CALL IMPDMP
      STOP
      END
```

FIGURE 9: Text Fortran Demo Program

Text output using resident courl0 font

Text output using TEX font cmr10.1200

There is a **bold** and an *italic* in this sentence

FIGURE 10: Output of Text Demo Program